EDA-Laboratorio 1

04/10/2013

Mikel Barcina
Jose Ángel Gumiel

Índice de contenido

1 Introducción

En este laboratorio se nos pide implementar un programa que pueda catalogar actores y su filmografía. La solución tendrá que ser eficiente, de forma que pueda cargar un fichero de texto que contenga una cantidad elevada de actores y películas, y que además se puedan realizar diversas operaciones y modificaciones sobre él.

Las operaciones que tendrá que hacer nuestro programa son las siguientes:

- Lectura de ficheros de texto.
- Búsqueda por actor que devuelva su filmografía.
- Organización de actores por orden alfabético.
- Posibilidad de añadir actores y películas manualmente.
- Una vez realizadas las operaciones, guardar fichero.

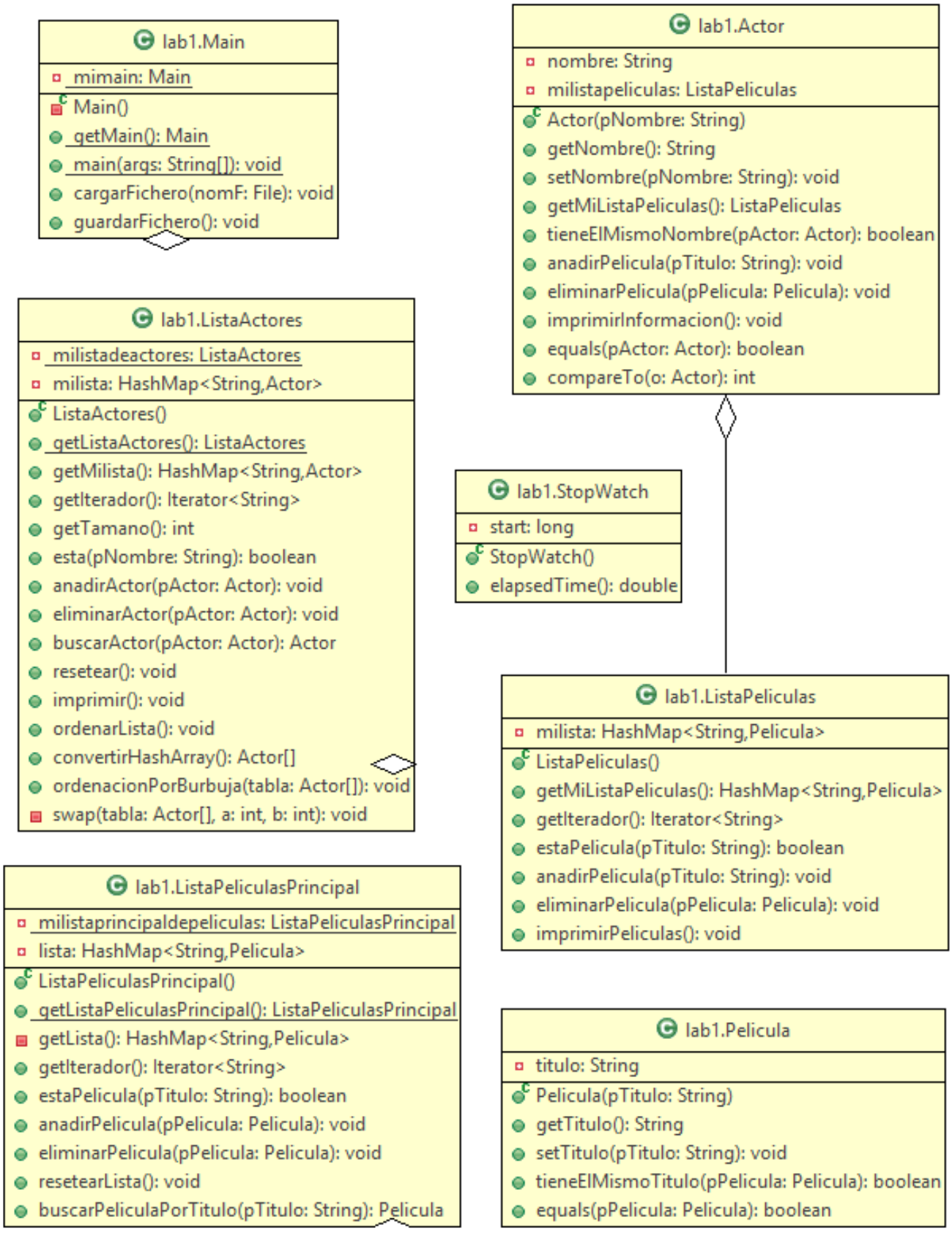
2 Diseño de las clases

Todas las clases que empleamos tienen los atributos privados y son accesibles mediante getters y setters.

En este laboratorio hemos usado 7 clases que detallamos a continuación:

- Main: Es la clase principal, la que ejecuta el programa. Destacar de esta clase que para la entrada y salida de datos hemos utilizado la clase JOptionPane, y sus métodos showMessageDialog() y showMessageInput() en vez de la clase Scanner que suele llevar a muchos errores y el System.out.println() que nos parecía demasiado simple para un trabajo de 2º año.
- Actor y Película: Son dos objetos con los que trabajaremos. Actor tiene nombre como identificador y también tiene el atributo miListaPeliculas. En los métodos tiene anadirPelicula, eliminarPelicula, imprimirInformacion y tieneElMismoNombre. Además de los getters y setters. Por otro lado, Película tiene el atributo título y el método tieneElMismoTítulo.
- ListaActores y ListaPelículas: Hemos usado los HashMaps. Llevan los métodos propios de listas, como el añadir, eliminar, esta e imprimir. La clase ListaActores es más completa, ya que incluye métodos para convertir el hashMap en Array, métodos para la ordenación (hemos usado el mergeSort) y buscarActor.
- ListaPeliculasPrincipal: Es una clase en la que se encuentra toda la colección de películas. Permite la búsqueda de películas.
- StopWatch: Clase creada para analizar los tiempos de ejecución.

En el siguiente diagrama de clases se pueden observar todas las clases mencionadas anteriormente y la totalidad de sus métodos y atributos, así como la relación existente entre ellas.


lab1.Main
mimain: Main
Main()
getMain(): Main
main(args: String[]): void
cargarFichero(nomF: File): void
guardarFichero(): void
lab1.Actor
nombre: String
milistapeliculas: ListaPeliculas
Actor(pNombre: String)
getNombre(): String
setNombre(pNombre: String): void
getMiListaPeliculas(): ListaPeliculas
tieneElMismoNombre(pActor: Actor): boolean
anadirPelicula(pTitulo: String): void
eliminarPelicula(pPelicula: Pelicula): void
imprimirInformacion(): void
equals(pActor: Actor): boolean
compareTo(o: Actor): int
lab1.ListaActores
milistadeactores: ListaActores
milista: HashMap<String,Actor>
ListaActores()
getListaActores(): ListaActores
getMilista(): HashMap<String,Actor>
getIterador(): Iterator<String>
getTamano(): int
esta(pNombre: String): boolean
anadirActor(pActor: Actor): void
eliminarActor(pActor: Actor): void
buscarActor(pActor: Actor): Actor
resetear(): void
imprimir(): void
ordenarLista(): void
convertirHashArray(): Actor[]
ordenacionPorBurbuja(tabla: Actor[]): void
swap(tabla: Actor[], a: int, b: int): void
lab1.StopWatch
start: long
StopWatch()
elapsedTime(): double
lab1.ListaPeliculas
milista: HashMap<String,Pelicula>
ListaPeliculas()
getMiListaPeliculas(): HashMap<String,Pelicula>
getIterador(): Iterator<String>
estaPelicula(pTitulo: String): boolean
anadirPelicula(pTitulo: String): void
eliminarPelicula(pPelicula: Pelicula): void
imprimirPeliculas(): void
lab1.ListaPeliculasPrincipal
milistaprincipaldepeliculas: ListaPeliculasPrincipal
lista: HashMap<String,Pelicula>
ListaPeliculasPrincipal()
getListaPeliculasPrincipal(): ListaPeliculasPrincipal
getLista(): HashMap<String,Pelicula>
getIterador(): Iterator<String>
estaPelicula(pTitulo: String): boolean
anadirPelicula(pPelicula: Pelicula): void
eliminarPelicula(pPelicula: Pelicula): void
resetearLista(): void
buscarPeliculaPorTitulo(pTitulo: String): Pelicula
lab1.Pelicula
titulo: String
Pelicula(pTitulo: String)
getTitulo(): String
setTitulo(pTitulo: String): void
tieneElMismoTitulo(pPelicula: Pelicula): boolean
equals(pPelicula: Pelicula): boolean

3 Descripción de las estructuras de datos principales

Para este laboratorio hemos usado las siguientes estructuras de datos para almacenar la información:

- HashMap: Es una estructura de datos claves (keys) con valores (values). La motivación para usar esta estructura es que ofrece un soporte muy eficiente para la búsqueda de datos. Se puede acceder a los elementos de la tabla con una clave generada. El funcionamiento del HashMap consiste en transformar la clave en un hash, un número que identifica la posición en la que se encuentra el valor deseado.

  Esta estructura gana utilidad cuando trabajamos con cantidades de información considerables, tal y como ocurre en nuestro caso.

- Array: Una estructura de datos ya conocida. Es un vector que almacena objetos de un mismo tipo. El Array lo usamos para poder trabajar con los algoritmos de ordenación, ya que las tablas hash no pueden ser ordenadas. Para ello usamos el método "convertirHashArray". Una vez que tenemos el contenido en el Array podemos proceder a la ordenación del mismo.

4 Diseño e implementación de los métodos principales

Para presentar los métodos principales los agruparemos por Clases:

Clase Main:

- Método main()

  Es el que ejecuta el programa principal. Consiste en un menú que nos permite escoger las opciones correspondientes. Damos 6 opciones:

```
1- Cargar fichero
2- Añadir un actor
3- Añadir una película
4- Ordenar la lista de actores
5- Guardar fichero
6- Salir
```

  Cada caso se trata de una manera independiente, llamando a los métodos y realizando las operaciones que se consideren oportunas.
  Sólo se podrá salir el programa cuando se seleccione la opción 6.

  Casos de prueba:

  - Introducir un número que no se encuentre entre el 1 y el 6.
  - Introducir un carácter en vez de un número.

- Método cargarFichero()

  Lanza una ventana que permite examinar la ubicación en la que tenemos guardado el archivo de texto. Una vez seleccionado, lo analiza y añade los actores y las películas a la lista.

  Coste: El algoritmo, carga todos los elementos del archivo. Para un archivo de n elementos, el coste será O(n).

  Casos de prueba:

  - Cargar un archivo pequeño y comprobar que la lista resultante se corresponde con el contenido del archivo de texto original.
  - Realizar la misma acción con un archivo más grande.
  - Probar con una lista vacía.

- Probar con un archivo que tenga un formato diferente a *.txt o que no exista.

- Método guardarFichero()

  Una vez de que tenemos actores y películas en nuestro programa, esta opción nos permite exportarlo a un fichero de texto. La ruta que hemos definido es C:/actrices.txt. Este método puede dar problemas, ya que es posible que se necesiten permisos por parte del SO para escribir en esa ubicación.

  Casos de prueba:

  - Añadir actores y seleccionar la opción para ver si se cumple.

Clase Actor:

- Método tieneElMismoNombre()

  Este método comprueba si dos actores tienen el mismo nombre o no y nos devuelve un boolean. En teoría no existen dos actores con el mismo nombre, luego, si encontramos que el resultado es true, será indicativo de que son la misma persona.

  Casos de prueba:

  Creamos tres actores, dos con nombres iguales y uno con nombre distinto y realizamos las siguientes comprobaciones:

  - assertTrue para el caso de que los dos tengan el mismo nombre.
  - assertFalse para el caso en el que comparamos nombres distintos.

- Método anadirPelicula()

  Añade una película a “milistapeliculas”, que es la lista de películas individual del actor.

  Casos de prueba:

  - Asegurar que la lista no es nunca null.
  - Comprobar que el tamaño cuando no añadimos ninguna película es 0. Añadir películas a la lista y comprobar cómo el tamaño de esta crece.

- Método eliminarPelicula()

  Elimina un objeto de tipo Película de “milisapeliculas”.

  Casos de prueba:

  - Eliminar una película que no se encuentra en la lista asegurándonos que se trata la excepción y que el programa continúa sin errores.
  - Añadir una película y asegurarnos que esa película se borra al introducir el comando eliminar. La comprobación se realiza por el tamaño de la lista, que se tiene que ver reducido y por el método imprimirInformacion(), de forma que nos aseguramos que la película que hemos eliminado es la correcta y no otra de la lista.

- Método imprimirInformacion()

  Muestra por pantalla el nombre del actor y su lista de películas.

  Casos de prueba:

  - Pedir que muestre una lista vacía.
  - Crear un Actor y añadir películas, introducir el comando para que las muestre por pantallas y comprobar manualmente que la relación se corresponde.

Clase Pelicula:

- Método tieneElMismoTitulo()

  Es el equivalente al "tieneElMismoNombre()" de actor. Comprueba si dos películas tienen el mismo título. No obstante, las pruebas a realizar serán del mismo tipo.

  No hay ningún otro método destacable en esta clase.

Clase ListaActores:

Esta clase contiene métodos que no consideramos de interés para explicar. Son anadirActor(), eliminarActor() e imprimir(). Son casi idénticos a otros que hemos explicado anteriormente, y se van a repetir también en la clase ListaPeliculas.

- Método esta()

  Comprueba si un actor está en la Lista de Actores. Esto se hace a través de las claves del HashMap.

  Casos de prueba:

  Dada una lista de actores no vacía haremos los siguientes tests:

  - Comprobar con un assertFalse al preguntar si un actor, que no está en la lista, está en la lista.
  - Comprobar con un assertTrue al preguntar si un actor, que sí que está en la lista, está en la lista.

- Método buscarActor()

  Comprueba si el Actor está en la lista y de ser así devuelve un objeto de tipo Actor. Si no está devuelve null.

  Casos de prueba:

  - Buscar un Actor que esté en la lista y comprobar que no es null.
  - Buscar un Actor que no esté en la lista y comprobar que es null.

- Método resetear()

  Vacía la lista dejándola sin ningún objeto.

  Casos de prueba:

  - Usar una lista no vacía, comprobar el tamaño, aplicar el método resetear y comprobar que el tamaño es cero.

- Método convertirHashArray()

  Como estamos trabajando con HasMaps, no podemos ordenar una estructura de datos de este tipo, así que tenemos que convertirla en Array para trabajar con ella y aplicar los algoritmos de ordenación.

- Método ordenacionPorBurbuja()

  Se utiliza para ordenar el Array. En el caso más desfavorable tiene **Coste $O(N^2)$.**

  Casos de prueba:

  Creamos un Array desordenado y lo imprimimos por pantalla. Aplicamos el algoritmo de

ordenación y volvemos a imprimirlo por pantalla. Comprobamos que lo haya imprimido en orden alfabético tras el uso del método.

- Método ordenarLista()

  Combina el método convertirHashArray() con ordenacionPorBurbuja().

Clase ListaPeliculas:

Los métodos que aparecen en esta clase ya han sido también utilizados en ListaActores, por lo que no hay que resaltar ni explicar ningún método nuevo.

Clase ListaPeliculasPrincipal:

Ocurre lo mismo que con la clase anterior. No hay ningún método nuevo, sólo los habituales para el manejo de listas.

Clase StopWatch:

- Método elapsedTime()

  Muestra el tiempo que ha transcurrido desde el momento inicial hasta que se para el tiempo.

5 Código

5.1 Programa

A continuación mostramos la totalidad del código que hemos implementado, ordenado por clases:

5.1.1 Clase Main

```
package lab1;

import java.io.*;  //Importo el Paquete Entero
import java.util.Iterator;
import javax.swing.JFileChooser;
import javax.swing.JOptionPane;

public class Main {
	//Atributos
	private static Main mimain = new Main();



	//Constructora
	private Main(){	}
```

```
	//Getters y Setters
	public static Main getMain(){
		return mimain;
	}


	//Otros Métodos

	public static void main(String[] args){
		boolean repetir;
		repetir = true;
		JOptionPane.showMessageDialog(null, "Bienvenido a la base virtual de cine");
		do{
			String entrada = JOptionPane.showInputDialog("¿Qué quieres hacer?\n"
					+ "1- Cargar fichero\n"
					+ "2- Añadir un actor\n"
					+ "3- Añadir una pelicula\n"
					+ "4- Ordenar la lista de actores\n"
					+ "5- Guardar fichero\n"
					+ "6- Salir");
			switch(entrada){
			case "1":
				JFileChooser fc = new JFileChooser();
				fc.setCurrentDirectory(new File("."));
				fc.setDialogTitle("Elige un fichero");
				fc.setAcceptAllFileFilterUsed(false);
				if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION)
{ //Comprueba que hemos seleccionado un archivo
					Main.getMain().cargarFichero(fc.getSelectedFile());
				} else {
					System.out.println("No seleccion ");
				}
				break;
			case "2":
				Actor anadir = new Actor(JOptionPane.showInputDialog("Introduce el
nombre del actor"));
				if(ListaActores.getListaActores().esta(anadir.getNombre())){
					System.out.println("El actor ya se encuentra en la lista");
//ListaActores lo comprueba también pero poniendo esto aqui te ahorras meterle peliculas para
nada.
				}else{
					boolean quedanpeliculas = true;
					String respuesta;
					do{
						respuesta = JOptionPane.showInputDialog("¿Quieres
añadirle una película? (Si/No)");
```

```
                                        if (respuesta.equalsIgnoreCase("si")){

        anadir.anadirPelicula(JOptionPane.showInputDialog("Introduce el titulo de la película"));
                                        }else if (respuesta.equalsIgnoreCase("no")){
                                                quedanpeliculas=false;
                                        }else{
                                                JOptionPane.showMessageDialog(null, "La
respuesta introducida es incorrecta. Por favor introduzca si o no");
                                        }
                                }while (quedanpeliculas);
                                ListaActores.getListaActores().anadirActor(anadir);
                        }
                        break;
                case "3":

        ListaPeliculasPrincipal.getListaPeliculasPrincipal().anadirPelicula(new
Pelicula(JOptionPane.showInputDialog("Introduce el titulo de la película")));
                        break;
                case "4":
                        ListaActores.getListaActores().ordenarLista();
                        break;
                case "5":
                        Main.getMain().guardarFichero();
                        break;
                case "6":
                        repetir=false;
                        break;
                default:
                        JOptionPane.showMessageDialog(null, "La opción itroducida es
incorrecta. Introduce un número del 1 al 5.");
                }
        }while(repetir==true);
        }



        public void cargarFichero(File nomF){
                try{
                         FileReader fr = new FileReader(nomF);
                         @SuppressWarnings("resource")
                        BufferedReader br = new BufferedReader(fr);
                         String linea, tituloaux;
                         Actor ultimoactor = null;
                         boolean todobien = true;
                         int ayuda = 0;
                         while ((linea=br.readLine())!=null) {
```

```
                                ayuda=0;

                                do{
                                        try{
                                                todobien=true;
                                                if(linea.length()!=0){
                                                        if(linea.substring(0,3).equals("\t\t\t")){
                                                                if(ultimoactor!=null){
                                                                        String[] sintabuladores =
linea.split("\t");
                                                                        String[] titulo =
sintabuladores[3-ayuda].split(" *[(]+\\d+[)]");
                                                                        tituloaux =
titulo[0].replaceAll("[\"]","");
                                                                        if(tituloaux.equals("")){
                                                                                todobien=false;
                                                                                ayuda++;
                                                                        }else{

ultimoactor.anadirPelicula(tituloaux);
                                                                        }
                                                                }else{

JOptionPane.showMessageDialog(null, "Ha ocurrido un error");
                                                                }
                                                        }else{
                                                                String[] division = linea.split("\t");
                                                                if(ayuda==0){
                                                                        ultimoactor = new
Actor(division[0]);
                                                                        if
(ListaActores.getListaActores().esta(division[0])){
                                                                                Actor aux = ultimoactor;
                                                                                ultimoactor =
ListaActores.getListaActores().buscarActor(aux);
                                                                        }else{

ListaActores.getListaActores().anadirActor(ultimoactor);
                                                                        }
                                                                }
                                                                String[] titulo = division[2-
ayuda].split(" *[(]+\\d+[)]");
                                                                tituloaux =
titulo[0].replaceAll("\"","");
                                                                if(tituloaux.equals("\t")){
                                                                        todobien=false;
                                                                        ayuda++;
                                                                }else{
```

```
ultimoactor.anadirPelicula(tituloaux);
                                                        }
                                                    }
                                                }
                                            }catch(ArrayIndexOutOfBoundsException ae){
                                                    //A veces en vez de salir actriz\t\tpelicula
solo hay un \t. Lo solucionamos aqui:
                                                    todobien=false;
                                                    ayuda++;

        ListaActores.getListaActores().eliminarActor(ultimoactor); //Para evitar duplicados
                                            }
                                    }while(!todobien);
                            }
                    }catch(IOException e) {
                            e.printStackTrace();
                    }
            }


            public void guardarFichero(){
                    FileWriter fichero = null;
             PrintWriter pw = null;
             try
             {
                 fichero = new FileWriter("C:\\actrices.txt");
                 pw = new PrintWriter(fichero);
                 String auxAct, auxPel;

                 Iterator<String> itrAct = ListaActores.getListaActores().getIterador();
                 Iterator<String> itrPel;

                 while(itrAct.hasNext()){
                   auxAct = itrAct.next();
                   pw.println(auxAct);
                   itrPel =
ListaActores.getListaActores().getMilista().get(auxAct).getMiListaPeliculas().getIterador();
                   while(itrPel.hasNext()){
                            auxPel=itrPel.next();
                            pw.println("--> "+auxPel);
                   }
                 }

             } catch (Exception e) {
                 e.printStackTrace();
```

```
        }

        finally {
           try {
                  if (null != fichero)
                     fichero.close();
           } catch (Exception e2) {
              e2.printStackTrace();
           }
        }
       }
}
```

5.1.2 Clase ListaActores

```
package lab1;

import java.util.HashMap;
import java.util.Iterator;
import javax.swing.JOptionPane;

public class ListaActores {
       //Atributos
       private static ListaActores milistadeactores = new ListaActores();
       private HashMap<String, Actor> milista;


       //Constructora
       public ListaActores(){
              milista = new HashMap<String, Actor>();
       }


       //Getters y Setters
       public static ListaActores getListaActores(){
              return milistadeactores;
       }

       public HashMap<String,Actor> getMilista(){
              return this.milista;
       }

       public Iterator<String> getIterador(){
              return this.getMilista().keySet().iterator();
```

```
}

public int getTamano(){
	return this.getMilista().size();
}


//Otros Metodos
	public boolean esta(String pNombre){
		return this.getMilista().containsKey(pNombre);
	}

	public void anadirActor(Actor pActor){
		try{
			if(esta(pActor.getNombre())){
				System.out.println("El actor ya se encuentra en la lista");
			}else{
				getMilista().put(pActor.getNombre(), pActor);
			}
		}catch(NullPointerException e){
			System.out.println("El actor que desea eliminar no existe");
		}
	}

public void eliminarActor(Actor pActor){
	try{
		if(esta(pActor.getNombre())){
			getMilista().remove(pActor);
		}else{
			System.out.println("El actor no se encuentra en la lista");
		}
	}catch(NullPointerException e){
		System.out.println("El actor que desea eliminar no existe");
	}
}

public Actor buscarActor(Actor pActor){
	try{
		if(esta(pActor.getNombre())){
			return this.getMilista().get(pActor.getNombre());
		}else{
			return null;
		}
	}catch(NullPointerException e){
```

```
                System.out.println("El actor que intentas buscar no existe");
                return null;
        }
}

public void resetear(){
        getListaActores().getMilista().clear();
}

public void imprimir(){
        Iterator<String> it = getListaActores().getIterador();
        while(it.hasNext()){
                getListaActores().getMilista().get(it.next()).imprimirInformacion();
                System.out.println("\n\n");  //Imprime dos lineas vacías.
        }
}

public void ordenarLista(){
        Actor[] milistaordenada = this.convertirHashArray();
        ordenacionPorBurbuja(milistaordenada);
        JOptionPane.showMessageDialog(null, "El resultado se muestra por consola");
        for(int i = 0; i<milistaordenada.length; i++)
                System.out.println(i+": "+milistaordenada[i].getNombre());
}

public Actor[] convertirHashArray(){
        Actor[] miArrayDeActores = new Actor[this.getMilista().size()];
        int i = 0;
        Iterator<String> itr = this.getIterador();
        while (itr.hasNext()){
                miArrayDeActores[i] = this.getMilista().get(itr.next());
                i++;
        }
        return miArrayDeActores;
}

public void ordenacionPorBurbuja(Actor[] tabla) {
        int out, in;
        for (out = tabla.length - 1; out > 0; out--)
                for (in = 0; in < out; in++)
                        if ( tabla[in].compareTo(tabla[in + 1]) > 0 )
                                swap(tabla, in, in + 1);
}
```

```
	private void swap(Actor[] tabla, int a, int b) {
		Actor aux = tabla[a];
		tabla[a] = tabla[b];
		tabla[b] = aux;
	}
}
```

5.1.3 Clase ListaPeliculasPrincipal

```
package lab1;

import java.util.HashMap;
import java.util.Iterator;

public class ListaPeliculasPrincipal {
	//Atributos
	private static ListaPeliculasPrincipal milistaprincipaldepeliculas = new
ListaPeliculasPrincipal();
	private HashMap<String, Pelicula> lista;


	//Constructora
	public ListaPeliculasPrincipal(){
		this.lista = new HashMap<String, Pelicula>();
	}


	//Getters y Setters
	public static ListaPeliculasPrincipal getListaPeliculasPrincipal(){
		return milistaprincipaldepeliculas;
	}


	private HashMap<String, Pelicula> getLista(){
		return this.lista;
	}

	public Iterator<String> getIterador(){
		return this.getLista().keySet().iterator();  //Esto crea un iterador de keys o
llaves

//en nuestro caso titulos de peliculas.
```

```
}


//Otros Metodos
public boolean estaPelicula(String pTitulo){
	return this.getLista().containsKey(pTitulo);
}

public void anadirPelicula(Pelicula pPelicula){
	if((pPelicula!=null)&&(!estaPelicula(pPelicula.getTitulo()))){
		this.getLista().put(pPelicula.getTitulo(), pPelicula);
	}else{
		System.out.println("la pelicula introducida no es valida.");
	}
}

public void eliminarPelicula(Pelicula pPelicula){
	if((pPelicula!=null)&&(estaPelicula(pPelicula.getTitulo()))){
		this.getLista().remove(pPelicula);
	}else{
		System.out.println("La pelicula no se encuentra en la lista.");
	}
}

public void resetearLista(){
	this.getLista().clear();
}

public Pelicula buscarPeliculaPorTitulo(String pTitulo){
	Pelicula rdo;
	try{
		if(estaPelicula(pTitulo)){
			rdo = this.getLista().get(pTitulo);
		}else{
			System.out.println("La pelicula no se encuentra en la lista");
			rdo=null;
		}
		return rdo;
	}catch(NullPointerException e){
		System.out.println("La película que intentas buscar no existe");
		return null;
	}
}
```

```
}
```

5.1.4 Clase ListaPeliculas

```
package lab1;

import java.util.HashMap;
import java.util.Iterator;

public class ListaPeliculas {
	//Atributos
	private HashMap<String, Pelicula> milista;

	//Constructora
	public ListaPeliculas(){
		this.milista = new HashMap<String, Pelicula>();
	}


	//Getter y Setters
	public HashMap<String, Pelicula> getMiListaPeliculas(){
		return this.milista;
	}

	public Iterator<String> getIterador(){
		return this.getMiListaPeliculas().keySet().iterator();  //Esto crea un iterador de
keys o llaves

//en nuestro caso titulos de peliculas.
	}


	//Otros Metodos
	public boolean estaPelicula(String pTitulo){
		try{
			return this.getMiListaPeliculas().containsKey(pTitulo);
		}catch(NullPointerException e){
			System.out.println("Estás tratando de añadir o eliminar una película no
válida");
			return false;
		}
	}
```

```
	public void anadirPelicula(String pTitulo){
		Pelicula aux;
		 if(pTitulo!=null){
			if(!this.estaPelicula(pTitulo)){

if(!ListaPeliculasPrincipal.getListaPeliculasPrincipal().estaPelicula(pTitulo)){
				aux = new Pelicula(pTitulo);

ListaPeliculasPrincipal.getListaPeliculasPrincipal().anadirPelicula(aux);
				this.getMiListaPeliculas().put(pTitulo,aux);
			 }else{
				aux                                                                    =
ListaPeliculasPrincipal.getListaPeliculasPrincipal().buscarPeliculaPorTitulo(pTitulo);
				this.getMiListaPeliculas().put(pTitulo,aux);
			 }
			}
		}
	}

	public void eliminarPelicula(Pelicula pPelicula){
		if(pPelicula!=null){
			if(estaPelicula(pPelicula.getTitulo())){
				this.getMiListaPeliculas().remove(pPelicula.getTitulo());
			}else{
				System.out.println("La película no se encuentra en la lista");
			}
		}
	}

	public void imprimirPeliculas(){
		Iterator<String> it = this.getIterador();
		while(it.hasNext()){
			System.out.println("->"+it.next());
		}
	}
}
```

5.1.5 Clase Actor

```
package lab1;
public class Actor{
	//Atributos
	private String nombre;
	private ListaPeliculas milistapeliculas;
```

```
//Constructora
public Actor(String pNombre){
	this.nombre=pNombre;
	this.milistapeliculas = new ListaPeliculas();
}

//Getters y Setters
public String getNombre() {
	return nombre;
}

public void setNombre(String pNombre) {
	this.nombre = pNombre;
}

public ListaPeliculas getMiListaPeliculas(){
	return this.milistapeliculas;
}

//Otros Metodos
public boolean tieneElMismoNombre(Actor pActor){
	try{
		if(pActor.getNombre().equals(this.getNombre())){
			return true;
		}else
			return false;
	}
	catch (Exception e){
		return false;
	}
}

public void anadirPelicula(String pTitulo){
	try{
	 this.getMiListaPeliculas().anadirPelicula(pTitulo);
	}catch(Exception e){
		System.out.println("La pelicula introducida no es válida");
	}
}

public void eliminarPelicula(Pelicula pPelicula){
	this.getMiListaPeliculas().eliminarPelicula(pPelicula);
}
```

```
	public void imprimirInformacion(){
		System.out.println("Nombre: "+this.getNombre());
		this.getMiListaPeliculas().imprimirPeliculas();
	}

	public boolean equals(Actor pActor){
		return this.tieneElMismoNombre(pActor);
	}

	public int compareTo(Actor o) {
		return this.getNombre().compareTo(o.getNombre());
	}
}
```

5.1.6 Clase Pelicula

```
package lab1;

public class Pelicula {
	//Atributos
	private String titulo;

	//Constructora
	public Pelicula(String pTitulo){
		this.titulo=pTitulo;
	}

	//Getters y Setters
	public String getTitulo(){
		return this.titulo;
	}

	public void setTitulo(String pTitulo){
		this.titulo=pTitulo;
	}

	public boolean tieneElMismoTitulo(Pelicula pPelicula){
		try{
			if(pPelicula.getTitulo().equals(this.getTitulo())){
				return true;
			}else
				return false;
		}
		catch (Exception e){
			return false;
```

```
		}
	}

	public boolean equals(Pelicula pPelicula){
		return this.tieneElMismoTitulo(pPelicula);
	}
}
```

5.1.7 Clase StopWatch

```
package lab1;

public class StopWatch {
	private final long start;
	public StopWatch(){
		start=System.currentTimeMillis();
	}


	public double elapsedTime(){
		long now = System.currentTimeMillis();
		return (now-start)/1000.0;
	}
```

5.2 Pruebas

En este apartado mostraremos las pruebas unitarias realizadas, clasificadas por clases.

5.2.1 Clase MainTest

```
package pruebas;

//import static org.junit.Assert.*;
import java.io.File;
import javax.swing.JFileChooser;
import lab1.Main;
import lab1.StopWatch;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class MainTest {
```

```
	@Before
	public void setUp() throws Exception {
	}

	@After
	public void tearDown() throws Exception {
	}

	@Test
	public void testCargarFichero() {
		JFileChooser fc = new JFileChooser();
		fc.setCurrentDirectory(new File("."));
		fc.setDialogTitle("Elige un fichero");
		fc.setAcceptAllFileFilterUsed(false);
		if (fc.showOpenDialog(null) == JFileChooser.APPROVE_OPTION) {
			System.out.println("Directorio: " + fc.getSelectedFile());
			StopWatch timer = new StopWatch();
			Main.getMain().cargarFichero(fc.getSelectedFile());
			System.out.println(timer.elapsedTime());
			Main.getMain().guardarFichero();
		} else {
			System.out.println("No seleccion");
		}
	}
}
```

5.2.2 Clase ListaActoresTest

```
package pruebas;

import static org.junit.Assert.*;
import lab1.Actor;
import lab1.ListaActores;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class ListaActoresTest {
	Actor actor1;
	Actor actor2;
	Actor actor3;
	Actor actor4;
	Actor actor5;
```

```
@Before
public void setUp() throws Exception {
	actor1 = new Actor("Robert Downey Jr");
	actor2 = new Actor("David Duchovny");
	actor3 = new Actor("Neil Patrick Harris");
	actor4 = new Actor("Chuck Norris");
}

@After
public void tearDown() throws Exception {
	actor1=null;
	actor2=null;
	actor3=null;
	actor4=null;
	actor5=null
	ListaActores.getListaActores().resetear();
}

@Test
public void testAnadirActor() {
	assertEquals(ListaActores.getListaActores().getTamano(),0);
	ListaActores.getListaActores().anadirActor(actor1);
	assertEquals(ListaActores.getListaActores().getTamano(),1);
	ListaActores.getListaActores().anadirActor(actor2);
	assertEquals(ListaActores.getListaActores().getTamano(),2);
	ListaActores.getListaActores().anadirActor(actor3);
	assertEquals(ListaActores.getListaActores().getTamano(),3);
	ListaActores.getListaActores().anadirActor(actor1);
	assertEquals(ListaActores.getListaActores().getTamano(),3);
	ListaActores.getListaActores().anadirActor(actor4);
	assertEquals(ListaActores.getListaActores().getTamano(),4);
	ListaActores.getListaActores().anadirActor(actor5);
	assertEquals(ListaActores.getListaActores().getTamano(),4);
}

@Test
public void testEliminarActor() {
	ListaActores.getListaActores().anadirActor(actor1);
	ListaActores.getListaActores().anadirActor(actor2);
	ListaActores.getListaActores().anadirActor(actor3);
	ListaActores.getListaActores().anadirActor(actor4);
	assertEquals(ListaActores.getListaActores().getTamano(),4);
	ListaActores.getListaActores().eliminarActor(actor1);
```

```
		assertEquals(ListaActores.getListaActores().getTamano(),3);
		ListaActores.getListaActores().eliminarActor(actor2);
		assertEquals(ListaActores.getListaActores().getTamano(),2);
		ListaActores.getListaActores().eliminarActor(actor2);
		assertEquals(ListaActores.getListaActores().getTamano(),2);
		ListaActores.getListaActores().eliminarActor(actor3);
		assertEquals(ListaActores.getListaActores().getTamano(),1);
		ListaActores.getListaActores().eliminarActor(actor4);
		assertEquals(ListaActores.getListaActores().getTamano(),0);
	}

	@Test
	public void testBuscarActor() {
		ListaActores.getListaActores().anadirActor(actor1);
		ListaActores.getListaActores().anadirActor(actor2);
		ListaActores.getListaActores().anadirActor(actor3);
		assertEquals(ListaActores.getListaActores().buscarActor(actor1),actor1);
		assertEquals(ListaActores.getListaActores().buscarActor(actor2),actor2);
		assertEquals(ListaActores.getListaActores().buscarActor(actor3),actor3);
		assertNull(ListaActores.getListaActores().buscarActor(actor4));

	}
}
```

5.2.3 Clase ListaPeliculasTest

```
package pruebas;

import static org.junit.Assert.*;
import lab1.ListaPeliculas;
import lab1.Pelicula;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class ListaPeliculasTest {
	Pelicula p1;
	Pelicula p2;
	Pelicula p3;
	Pelicula p4;
	ListaPeliculas lp;
```

```
@Before
public void setUp() throws Exception {
	p1 = new Pelicula("Iron Man");
	p2 = new Pelicula("How High");
	p3 = new Pelicula("Thor");
	lp = new ListaPeliculas();
}

@After
public void tearDown() throws Exception {
	p1=null;
	p2=null;
	p3=null;
	p4=null;
	lp=null;
}

@Test
public void testAnadirPelicula() {
	assertFalse(lp.estaPelicula(p1.getTitulo()));
	assertFalse(lp.estaPelicula(p2.getTitulo()));
	assertFalse(lp.estaPelicula(p3.getTitulo()));
	lp.anadirPelicula(p1.getTitulo());
	assertTrue(lp.estaPelicula(p1.getTitulo()));
	assertFalse(lp.estaPelicula(p2.getTitulo()));
	assertFalse(lp.estaPelicula(p3.getTitulo()));
	lp.anadirPelicula(p2.getTitulo());
	assertTrue(lp.estaPelicula(p1.getTitulo()));
	assertTrue(lp.estaPelicula(p2.getTitulo()));
	assertFalse(lp.estaPelicula(p3.getTitulo()));
	lp.anadirPelicula(p3.getTitulo());
	assertTrue(lp.estaPelicula(p1.getTitulo()));
	assertTrue(lp.estaPelicula(p2.getTitulo()));
	assertTrue(lp.estaPelicula(p3.getTitulo()));
	assertTrue(lp.estaPelicula(p1.getTitulo()));
	assertTrue(lp.estaPelicula(p2.getTitulo()));
	assertTrue(lp.estaPelicula(p3.getTitulo()));
	//Es imposible pasarle null a este método
}

@Test
public void testEliminarPelicula()  {
	lp.anadirPelicula(p1.getTitulo());
	lp.anadirPelicula(p2.getTitulo());
```

```
		lp.anadirPelicula(p3.getTitulo());
		assertTrue(lp.estaPelicula(p1.getTitulo()));
		assertTrue(lp.estaPelicula(p2.getTitulo()));
		assertTrue(lp.estaPelicula(p3.getTitulo()));
		lp.eliminarPelicula(p1);
		assertFalse(lp.estaPelicula(p1.getTitulo()));
		assertTrue(lp.estaPelicula(p2.getTitulo()));
		assertTrue(lp.estaPelicula(p3.getTitulo()));
		lp.eliminarPelicula(p2);
		assertFalse(lp.estaPelicula(p1.getTitulo()));
		assertFalse(lp.estaPelicula(p2.getTitulo()));
		assertTrue(lp.estaPelicula(p3.getTitulo()));
		lp.eliminarPelicula(p3);
		assertFalse(lp.estaPelicula(p1.getTitulo()));
		assertFalse(lp.estaPelicula(p2.getTitulo()));
		assertFalse(lp.estaPelicula(p3.getTitulo()));
		//Como es imposible añadir una pelicula=null al método anterior, a este también
	}


	@Test
	public void testEstaPelicula(){
		lp.anadirPelicula(p1.getTitulo());
		lp.anadirPelicula(p2.getTitulo());
		assertTrue(lp.estaPelicula(p1.getTitulo()));
		assertTrue(lp.estaPelicula(p2.getTitulo()));
		assertFalse(lp.estaPelicula(p3.getTitulo()));
		assertFalse(lp.estaPelicula(p4.getTitulo()));
	}
}
```

5.2.4 Clase ActorTest

```
package pruebas;

import static org.junit.Assert.*;
import lab1.Actor;
import lab1.Pelicula;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;
```

```
public class ActorTest {
	Actor actor1;
	Actor actor2;
	Actor actor3;
	Actor actor4;
	Actor actor5;
	Pelicula p1;
	Pelicula p2;

	@Before
	public void setUp() throws Exception {
		actor1 = new Actor("Robert Downey Jr");
		actor2 = new Actor("David Duchovny");
		actor3 = new Actor("Neil Patrick Harris");
		actor4 = new Actor("Chuck Norris");
		p1 = new Pelicula("Avengers");
		p2 = new Pelicula("X-Files");
	}

	@After
	public void tearDown() throws Exception {
		actor1=null;
		actor2=null;
		actor3=null;
		actor4=null;
		actor5=null;
		p1=null;
		p2=null;
	}

	@Test
	public void testTieneElMismoNombre() {
		assertFalse(actor1.tieneElMismoNombre(actor2));
		assertFalse(actor1.tieneElMismoNombre(actor3));
		assertFalse(actor1.tieneElMismoNombre(actor4));
		actor2.setNombre("Robert Downey Jr");
		assertTrue(actor1.tieneElMismoNombre(actor2));
		assertFalse(actor3.tieneElMismoNombre(actor5));
	}

	@Test
	public void testAnadirPelicula()  {
		assertFalse(actor1.getMiListaPeliculas().estaPelicula(p1.getTitulo()));
		assertFalse(actor1.getMiListaPeliculas().estaPelicula(p2.getTitulo()));
```

```
            actor1.anadirPelicula(p1.getTitulo());
            assertTrue(actor1.getMiListaPeliculas().estaPelicula(p1.getTitulo()));
            assertFalse(actor1.getMiListaPeliculas().estaPelicula(p2.getTitulo()));
            actor1.anadirPelicula(p2.getTitulo());
            assertTrue(actor1.getMiListaPeliculas().estaPelicula(p1.getTitulo()));
            assertTrue(actor1.getMiListaPeliculas().estaPelicula(p2.getTitulo()));
        }

        @Test
        public void testEliminarPelicula()  {
            actor1.anadirPelicula(p1.getTitulo());
            actor1.anadirPelicula(p2.getTitulo());
            assertTrue(actor1.getMiListaPeliculas().estaPelicula(p1.getTitulo()));
            assertTrue(actor1.getMiListaPeliculas().estaPelicula(p2.getTitulo()));
            assertTrue(actor1.getMiListaPeliculas().estaPelicula(p1.getTitulo()));
            assertTrue(actor1.getMiListaPeliculas().estaPelicula(p2.getTitulo()));
            actor1.eliminarPelicula(p2);
            assertTrue(actor1.getMiListaPeliculas().estaPelicula(p1.getTitulo()));
            assertFalse(actor1.getMiListaPeliculas().estaPelicula(p2.getTitulo()));
            actor1.eliminarPelicula(p1);
            assertFalse(actor1.getMiListaPeliculas().estaPelicula(p1.getTitulo()));
            assertFalse(actor1.getMiListaPeliculas().estaPelicula(p2.getTitulo()));
        }
}
```

6 Conclusiones

En este laboratorio hemos aprendido a utilizar estructuras de datos que hasta ahora desconocíamos, como es el caso del HashMap, que nos permiten organizar la información de una forma más eficiente, reduciendo así los costes del programa.

También hemos utilizado nuevas funciones, como la pequeña interfaz que permite seleccionar la operación a realizar y que además permite buscar el archivo a añadir sin tener que teclear la ruta en la que se encuentra.